Achievements

Spreadsheets are important, but users do not see themselves as programmers, so they suffer from similar problems like complexity and lack of documentation.

To address lack of documentation, we tried reverse engineering:

* Class diagrams
* Dataflow visualization

To address complextiy:

* Metrics
* Smell detection
* Refactoring
* Clone detection
* Testing

This all has to do with understanding and improving *quality* we still want to support understanding domain/calculation.Our work and that of others was all 100% automated, what we have learned, if we want to continue with reverse engineering, we need more support from the user, both the creator and human level intelligence (a generic user): we both want to do large studies with generic users like the labeling game and make support for users to add more (domain) knowledge to the sheets.

Challenges

* They don’t see themselves as programmers
  + Most users do not have proper trainer
  + The solution is not to give them tools and training (make them see they are programmers), they should remain seeing themselves as non-programmers, but with fewer errors/more expressive power.
  + Lack of best practices/programming standards, which leads to a variety in different spreadsheet types.
* Lack of data (broader issue in very applied/industrial research, tradeoff realism vs. reproducability)
  + When we get large sets, we do not have the users with them
  + No process data (version control history, like “MSR type papers”)
* Spreadsheets can get big, navigating on 1 worksheet is still ok, but for larger sheets it is hard to filter what to present, just showing all dependencies is too much. [example from Enron, Sohon might have a pic]
  + We have to leave the cell level, want to look at the computation level (slicing)
  + Getting domain knowledge out at a higher level, maybe with some assistance from the user, letting them know which cells have labels and which lack them.
  + Performance for users (Allaeddin)
* Lack of open standards, and even when open, not very standardazed (we have to reverse engineer stuff, link SCAM, also HPC/HEAT is not open, not willing to share)
* Limited options for customization (undo/redo stack is not available, in the object model precedents do not go deeper than 1 level, objects model gets extended but not updated)

Future opportunities

* Continuation of the refactoring and testing work, in contracts with the smells which have been tried in practice and picked up by a larger research community, refactorings need to be made more applicable.
* More deeply understanding the role of spreadsheets in the enterprise, both culturally (evolution, process) and in their application (use of spreadsheets) For example, classification different types of spreadsheets, potentially. For this we are exploring the following concrete endeavours:
  + For the purpose of reverse engineering (Sohon)
    - Potential application of extraction high level structure would be
      * documentation extraction
      * validating spreadsheets at a higher level
  + To support users in creating better spreadsheets with an alternative interface for some types of spreadsheets, bringing a higher level of abstraction (Bas)
* Beyond formulas. Self-service BI. This field is now mainly concerned with building but not with maintenance. We see them making the same mistake by not focusing on end-user maintenace, maybe underestimating the longevity of the new artefacts created. We suspect that this new incarnation of end-user programming again will suffer from smells etc. So our research will explore this, for example by focusing. Other spreadsheet concepts like PivotTables and VBA code
  + Increasing performance of spreadsheets (Allaeddin)
  + Intertwining of formulas
  + This makes automated analysis even harder